ChatGPT

Refactoring HealthPrep for Multi-Specialty Protocols

Current Design Assumptions (Single Specialty per Client ID)

Organization = One Practice/Specialty: In the current **HealthPrepV2** design, each **Client ID** (organization) is treated as a distinct medical practice with a single patient population and specialty focus. The `Organization` model even includes a `specialty` field (e.g. *"Family Practice"* or *"Cardiology"*) to label the practice's specialty [1]. All key data models (patients, documents, screenings, etc.) are scoped by `org_id` to isolate data per organization [2]. This means **screening protocols (ScreeningTypes)** are defined at the organization level and shared by all users/providers in that org.

> **Code evidence:** Each screening type is tied to an organization via `org_id` [2], and when the screening engine runs, it **loads all active screening types for the patient's organization** (no further filtering by provider or sub-group) [3]. In fact, the engine explicitly notes "only get screening types from patient's organization," confirming that **the system currently assumes one uniform protocol per org** [3].

No Per-Provider Customization: There is currently no concept of per-provider or per-department variation in screening criteria within an organization. All providers under the same client/org share the exact same checklist protocol. For example, a cardiologist and an oncologist in the same organization would see the same list of screenings (assuming the screenings have broad eligibility). The code does not track provider specialty for each screening or patient; the Appointment model simply stores a provider name as a string, with no link to a specialty or separate provider profile [4]. The **appointment prioritization** and screening refresh logic operate at the org level – e.g. fetching upcoming appointments by org_id and then updating screenings for those patients [5] [3]. Nowhere does it filter or branch logic by which provider will see the patient, only by whether the patient is in the org and (optionally) has an appointment soon.

Preset Protocols Grouped by Specialty: HealthPrep does support *specialty-specific screening presets* as templates. The system comes with preset bundles for various specialties (cardiology, oncology, primary_care, etc.) that can be imported. Each `ScreeningPreset` has a `specialty` attribute (e.g. `"cardiology"`, `"oncology"`) to identify its domain [6]. These presets are intended to provide a ready-made set of screening types appropriate for that specialty, which an admin can apply to their organization. However, once applied, the screening types become part of the organization's global set. The UI makes it clear that applying a preset will **"make all screening types from this preset available to all users in your organization."** [7] There is no built-in mechanism to limit certain screening types to only some providers within the org.

Effect of Current Design: Under these assumptions, **one organization = one protocol.** This works fine if each organization is a single-specialty practice (the original intent). The organization's `specialty` field can be used to choose a preset to import, and all patients in that org are evaluated against that one set of screening criteria. The software prioritizes patients with upcoming appointments (again, at the org level)

and generates a **Quality Checklist** that lists each due or completed screening for the patient [8]. Because all screenings are shared org-wide, the **quality checklist will include every eligible screening type for that patient**, even if some screenings are irrelevant to the specific provider's specialty.

Example: If an org imported both a cardiology and an oncology preset under one Client ID, a patient who is an elderly female might end up with screenings for both cardiology (e.g. lipid panel, EKG) and oncology (e.g. mammogram, colonoscopy) in her record. **Currently, the prep sheet's checklist would show *all* of those** for that patient, since it does `Screening.query.filter_by(patient_id=…)` with no further filter [9]. There is no awareness of whether the upcoming appointment is with a cardiologist or oncologist – the checklist is universal. This is the crux of the problem: **different providers/specialties have different chart prep needs, but the software treats them uniformly if they share an org (Client ID).**

Why This Is a Problem for Multi-Specialty Practices

If a multi-specialty clinic uses a single HealthPrep client/org for all providers, they face two suboptimal choices:

- **Use one unified protocol for all:** They might only apply one preset (say, "Primary Care") to cover general screenings. This means specialists (cardiology, oncology, etc.) won't have their specific screening needs addressed, or conversely, if a broad preset is used, it may contain many screenings irrelevant to certain specialists. In our example, a cardiologist doesn't typically manage cancer screenings, and an oncologist might not focus on cholesterol checks – yet the unified protocol would show both categories to each.

- **Apply multiple specialty presets to the org:** HealthPrep allows importing multiple presets (cardiology, oncology, etc.) into one organization. However, all those screening types then coexist in the database for that org. **There is no built-in per-provider segregation**, so a patient will be evaluated for *every* screening type across all imported presets (provided they meet gender/age criteria). This could overwhelm the prep sheet with irrelevant items for a given provider. For instance, a cardiology appointment prep might list oncology screenings that the cardiologist isn't going to act on, simply because those screenings exist in the system and the patient qualifies for them. The **software currently cannot distinguish context** – it assumes the org's entire screening list is always relevant.

In summary, **multi-specialty clinics currently cannot tailor the prep checklist by provider.** All providers under the same Client ID see the same "quality checklist" content. This undermines the goal of providing "specific protocols for the needs of each individual practice/provider," as the user noted. It can also reduce efficiency or usefulness of the prep sheet, since providers may have to sift through unrelated screening reminders.

Refactoring Approach Overview

To support per-provider or per-specialty customization within a single organization, we need to introduce a new level of **protocol segmentation** below the organization level. Broadly, two approaches were considered:

Option 1 – Split into Multiple Organizations: Treat each specialty (or even each provider) as a separate client/organization in the system, each with its own Client ID and screening protocol configuration. While this would technically isolate protocols, it's **not ideal and likely unworkable** in practice: - *Single EHR Integration:* Each `Organization` in HealthPrep has its own Epic FHIR API credentials and connection settings [10]. A multi-specialty clinic typically uses one set of Epic credentials for the whole practice, so splitting into multiple orgs would require duplicating or sharing credentials across orgs (which the system may not support easily) or registering each specialty as a separate client with the EHR – an unnecessary complication [10]. - *Data fragmentation:* Patients and data would be siloed per org. The same patient seeing two specialists would exist twice in the database (once per specialty-org), defeating the purpose of a unified view. It would also complicate syncing – the system might pull the same patient's records twice for two orgs. - *User management:* Admins and staff would have to manage multiple logins or contexts for one clinic, which is cumbersome.

For these reasons, splitting one real-world clinic into multiple logical orgs is **not the recommended solution**. HealthPrep's multi-tenancy is meant for completely separate clients (e.g. different clinics) [11], not for subdivisions of one clinic.

Option 2 – Support Multi-Specialty Inside One Organization: This is the recommended path. We will **refactor the data model and logic to handle multiple distinct protocols within a single org**, keyed by provider or specialty. The goal is to allow, for example, a cardiology protocol and an oncology protocol to coexist in one organization, but **only surface the relevant one for a given provider's patients**. Key components of this refactor include:

- **Provider Specialty Mapping:** Introduce a way to identify each provider's specialty in the system, so we know which protocol to use for their appointments. Currently, appointments only store the provider name as text [4], so we need to augment this:
- *Provider Model:* Create a new `Provider` (or similar) model/table for the organization, with fields like `name`, `specialty`, and perhaps an external ID if needed. This model would represent the physicians/practitioners. Each Provider belongs to an Organization and has a declared specialty. We can populate this table either via an admin interface or even during appointment import (e.g. on first seeing a provider name, create a record and let an admin set the specialty).
- *Link Appointments to Provider:* Change the `Appointment` model to reference `Provider` (foreign key) instead of or in addition to the provider name string. This might involve a migration where we match existing appointments' provider names to new Provider entries. Going forward, each appointment will then inherently carry the provider's specialty via this link.

- *Alternative:* If introducing a full Provider model is too heavy, a simpler (but less normalized) approach is to add a `specialty` field to the `Appointment` itself. For example, if the upstream scheduling system provides an appointment type or department that indicates specialty, we could store that. However, a Provider model is more robust and avoids repeating data on every appointment.

- **Tag Screening Types by Specialty:** We need to mark which screening types belong to which protocol/specialty. Currently, once presets are applied, all `ScreeningType` entries are just lumped under the org with no further classification. We will add a `specialty` attribute to the ScreeningType model:

- This can be a string field (holding values like `"cardiology"`, `"oncology"`, etc., matching the preset names), or a foreign key linking to a Specialty/Department table (if we want to formalize specialties).
- When importing/applying a preset to an org, the code should set the `specialty` field on each new ScreeningType created. For example, if applying the Cardiology preset, every ScreeningType from that preset gets `specialty="cardiology"`. The preset metadata already carries a specialty name [12], so we can propagate that easily.
- For existing screening types in single-specialty orgs, we can retroactively assign the org's specialty to them (or default to `"general"` if specialty doesn't neatly apply). This migration ensures no null specialties unless intended.

- We should also adjust uniqueness constraints: previously, the system likely assumed screening type names are unique per org. With a new specialty field, we might allow the same name to exist under different specialties in one org (if that ever happens). We can enforce uniqueness on `(org_id, name, specialty)` if needed. (If there are common screenings across specialties – e.g. "Flu Shot" – we could treat them as "general" or assign them to one specialty. But allowing duplicates by specialty might be cleaner for isolation.)

- **Multiple Protocols Coexistence Logic:** With screening types labeled by specialty, we can now filter which ones to consider for a given context:

- **When generating a Prep Sheet for an appointment:** We will determine the provider's specialty from the appointment (via the linked Provider or a stored field). Then, we will filter the quality checklist to only include screenings of that specialty (plus any general ones that apply to everyone, if we define some as general).
    - Concretely, in `PrepSheetGenerator._generate_quality_checklist`, instead of pulling **all** screenings for the patient [9], we add a specialty filter. For example:

      ```
      if appointment:
          specialty = appointment.provider.specialty
          screenings = Screening.query.filter_by(patient_id=patient_id) \
                        .join(ScreeningType) \
                        .filter(ScreeningType.specialty == specialty).all()
      else:
          screenings =
      Screening.query.filter_by(patient_id=patient_id).all()
      ```

      This way, when generating a prep sheet in the context of a specific appointment (e.g. via `/prep_sheet/appointment/<id>` route), the checklist will only include the screenings relevant to that provider's specialty. A cardiologist's prep sheet will show cardiology screenings (and omit, say, Pap smear), whereas an oncologist's prep sheet will show the cancer-related screenings, etc.
    - We must ensure that *general* screenings (if any) that truly apply to all providers are handled. One approach is to designate a specialty value like `"general"` for such screenings and always include those regardless of provider. For example, basic preventive screenings might be shown to all providers as FYI. This is a product decision – do we want strictly specialty-

specific lists, or some overlap? We can configure the filter accordingly (e.g., `ScreeningType.specialty IN (specialty, 'general')`).

- **When refreshing screenings (engine logic):** The `ScreeningEngine.refresh_patient_screenings()` currently iterates through all screening types for the org [3] and creates/updates `Screening` records for those the patient is eligible for. We might leave this as-is for now, meaning a patient will have screening records for all specialties' types if they meet criteria. This ensures the data is prepared. It does extra work (some screenings might never be shown if the patient never sees that specialist), but it keeps things simpler and ensures if a patient later has a new specialty appointment, we don't have to retroactively generate records.
    - An optimization later could be to only generate screenings for specialties the patient has encountered. For instance, if a patient has only ever seen cardiology, maybe we don't need to compute oncology screenings until they get an oncology appointment or condition. However, implementing that conditional generation is complex (we'd need to detect specialty context in the engine). Given that eligibility criteria (gender/age/conditions) already filter out many irrelevancies, the performance hit may be acceptable. We can revisit optimization after functional correctness is achieved.

- **In the UI screening lists:** If there is an admin or staff view that lists all screenings or patients, we might introduce filters or grouping by specialty. For example, an admin could toggle to see "cardiology protocol" vs "oncology protocol" compliance. This would be a new feature – currently, the interface doesn't distinguish, but adding a filter dropdown by specialty on the screening list page would be helpful in a multi-specialty org. This can reuse the `ScreeningType.specialty` field to group items.

- **Migration and Backward Compatibility:** We will need to carefully migrate existing data and configuration:

- **Existing Single-Specialty Orgs:** For organizations that have only one specialty (which is likely all current ones, since the feature was built with that in mind), we can auto-fill the new `ScreeningType.specialty` for all their screening types using the org's known specialty. For example, if Org 5 is labeled as "Cardiology" [1], every screening_type in Org 5 gets `specialty="Cardiology"`. This ensures no change in behavior (the prep sheets will still show all those screenings, and if we filter by that specialty it's the same set).
- **Organizations that applied multiple presets already:** If any orgs have (perhaps experimentally) imported more than one preset, we need to assign specialties to those screening types. If the presets are still identifiable (the `ScreeningPreset` records might exist with `org_id` and `specialty`), we can match screening types to their preset of origin. For example, by matching names or using the preset's `screening_data`. If that's difficult, an admin may have to manually categorize some screening types. We can provide an interface in the admin panel to edit the specialty of each screening type in an org, just for the initial cleanup. It's likely this scenario is rare at this stage.
- **Database changes:** Add the new tables/columns (Provider model, `ScreeningType.specialty`). Ensure foreign keys and indexes are set (e.g., index on `ScreeningType.specialty` if we will filter by it often). Also update any unique constraints on `ScreeningType` as mentioned. Migration scripts should populate the specialty field as above for existing data.

- **Testing:** After refactoring, test with various scenarios:
    - Single-specialty org (should behave exactly as before).
    - Multi-specialty org with two presets applied: verify that prep sheet for a provider of specialty A shows only A's screenings and vice versa. Test a patient who qualifies for both sets – ensure filtering works.
    - Patients with no appointments: the screening list in the admin UI might show all screenings (that's fine, since it's a comprehensive view). But if needed, implement a filter.
    - Appointment-based prioritization: ensure that still runs (it will queue patients by org as before). It will trigger screening refresh for the patient (which generates all types). The prep sheet generation is where we actually tailor the output, so prioritization logic mostly remains unchanged.
- **Guidance for Usage:** With the changes, documentation to users should clarify:
- When an organization has multiple specialties, they should **apply the corresponding presets** for each specialty they need. The system will then maintain all those screenings internally, but will separate them by context when generating prep sheets.
- Ensure that each provider in the system is correctly labeled with a specialty (so that the filtering works). Perhaps in the UI's user management or a new provider management section, list providers and let the admin set their specialty if not auto-set.
- If a clinic adds a new specialty later, they can import the new preset and assign it to the relevant provider. The system will integrate the new screenings and use them for that provider's patients going forward.

By implementing the above, **HealthPrep will gain a true multi-specialty capability**. A single Client ID (org) can encompass a multi-specialty clinic, yet each provider will effectively have their own screening protocol. For example, cardiology appointments will get a cardiology-specific quality checklist, oncology appointments an oncology-specific checklist, etc., all while sharing the same patient database and Epic integration. This refactoring aligns the software with the real-world scenario where one practice/group might host multiple specialties under one roof.

Conclusion

In summary, the software was originally built under the assumption of "one organization = one specialty protocol" [1] [3]. To accommodate multiple protocols under the same client, we recommend enhancing the data model to record provider specialties and tagging screening types with a specialty. The prep sheet generation logic will then filter or select the appropriate checklist items based on the provider or specialty context. This refactor (along with minor UI additions for managing and viewing specialties) will allow a cardiologist and an oncologist in the same organization to use **different chart-prepping checklists**, as intended.

By focusing on this refactoring first, we ensure the foundation supports per-provider specialization. Subsequent features (like perhaps provider-specific custom tweaks, or reporting by specialty) can be built on this. The end result will be a more flexible HealthPrep system that truly **"works with specific protocols for the needs of each individual practice/provider"** as required, without forcing clinics to split into separate client IDs.

Sources:

- HealthPrepV2 data model and multi-tenancy design [1] [2] [10]
- Screening engine and preset application logic in HealthPrepV2 [3] [7] [9]
- Appointment and provider data structures [4] and specialty preset definitions [6].

---

[1] [2] [4] [6] models.py
https://github.com/mitfusco98/HealthPrepV2/blob/ff3be9d263e1e36af59641c220b80c1e9e15afca/models.py

[3] engine.py
https://github.com/mitfusco98/HealthPrepV2/blob/ff3be9d263e1e36af59641c220b80c1e9e15afca/core/engine.py

[5] appointment_prioritization.py
https://github.com/mitfusco98/HealthPrepV2/blob/ff3be9d263e1e36af59641c220b80c1e9e15afca/services/appointment_prioritization.py

[7] presets.html
https://github.com/mitfusco98/HealthPrepV2/blob/ff3be9d263e1e36af59641c220b80c1e9e15afca/templates/admin/presets.html

[8] [9] generator.py
https://github.com/mitfusco98/HealthPrepV2/blob/ff3be9d263e1e36af59641c220b80c1e9e15afca/prep_sheet/generator.py

[10] [11] MULTI_TENANCY_SETUP.md
https://github.com/mitfusco98/HealthPrepV2/blob/ff3be9d263e1e36af59641c220b80c1e9e15afca/MULTI_TENANCY_SETUP.md

[12] load_specialty_presets.py
https://github.com/mitfusco98/HealthPrepV2/blob/ff3be9d263e1e36af59641c220b80c1e9e15afca/load_specialty_presets.py